To understand the sequence of integers printed by the call `ex233(6)`, let's break down the function step-by-step. The function works recursively and prints `n` at the beginning and end of each call, with recursive calls to `ex233(n-2)` and `ex233(n-3)` in between.

Here is the function for reference:

```java
public static void ex233(int n)
{
    if (n <= 0) return;
    StdOut.println(n);
    ex233(n-2);
    ex233(n-3);
    StdOut.println(n);
}
```

Let's trace the call `ex233(6)`:

1. `ex233(6)`:
   - Print `6`
   - Call `ex233(4)`
2. `ex233(4)`:
   - Print `4`
   - Call `ex233(2)`
3. `ex233(2)`:
   - Print `2`
   - Call `ex233(0)` (returns immediately because `n <= 0`)
   - Call `ex233(-1)` (returns immediately because `n <= 0`)
   - Print `2`
   - Return to `ex233(4)`
4. Back in `ex233(4)`:
   - Call `ex233(1)`
5. `ex233(1)`:
   - Print `1`
   - Call `ex233(-1)` (returns immediately because `n <= 0`)
   - Call `ex233(-2)` (returns immediately because `n <= 0`)
   - Print `1`
   - Return to `ex233(4)`
6. Back in `ex233(4)`:
   - Print `4`
   - Return to `ex233(6)`
7. Back in `ex233(6)`:

   - Call `ex233(3)`
8. `ex233(3)`:
   - Print `3`
   - Call `ex233(1)`
9. `ex233(1)`:
   - Print `1`
   - Call `ex233(-1)` (returns immediately because `n <= 0`)
   - Call `ex233(-2)` (returns immediately because `n <= 0`)
   - Print `1`
   - Return to `ex233(3)`
10. Back in `ex233(3)`:
    - Call `ex233(0)` (returns immediately because `n <= 0`)
    - Print `3`
    - Return to `ex233(6)`
11. Back in `ex233(6)`:
    - Print `6`
    - End

Combining all the prints in order:

1. 6
2. 4
3. 2
4. 2
5. 1
6. 1
7. 4
8. 3
9. 1
10. 1
11. 3
12. 6

Thus, the sequence of integers printed by the call `ex233(6)` is:

```
6 4 2 2 1 1 4 3 1 1 3 6
```